Covid Data Analysis

Load

Load libraries, create variables, and preprocess the data

```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv("Covid Data.csv")
df["DEATH"] = (df["DATE_DIED"] != "9999-99-99").astype(int)
X = df.drop(columns=["DATE_DIED", "DEATH"])
y = df["DEATH"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
accuracy_list, precision_list, recall_list, f1_list = [], [], [], []
```

Analysis

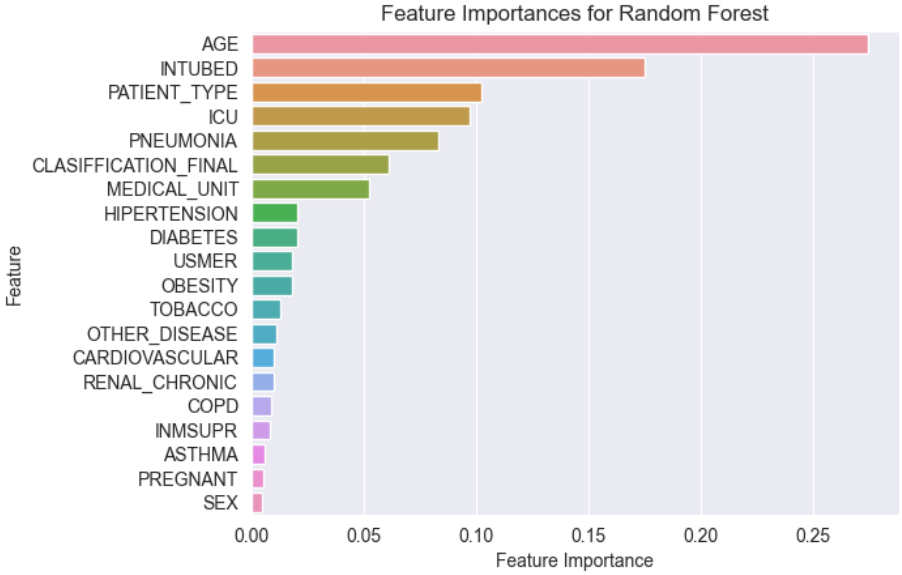
Feature Analysis

We begin with a random forest feature analysis.

```
# Create random forest classifier
forest = RandomForestClassifier(n_estimators=110, random_state=42, max_depth=60)
forest.fit(X_train, y_train)
```

```
RandomForestClassifier(max_depth=60, n_estimators=110, random_state=42)
```

```
importances = forest.feature_importances_
feature_names = X.columns
indices = np.argsort(importances)[::-1]
sorted_feature_names = [feature_names[i] for i in indices]
sns.barplot(x=importances[indices], y=sorted_feature_names)
plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.title("Feature Importances for Random Forest")
plt.savefig("feature_importances.png")
plt.show()
```

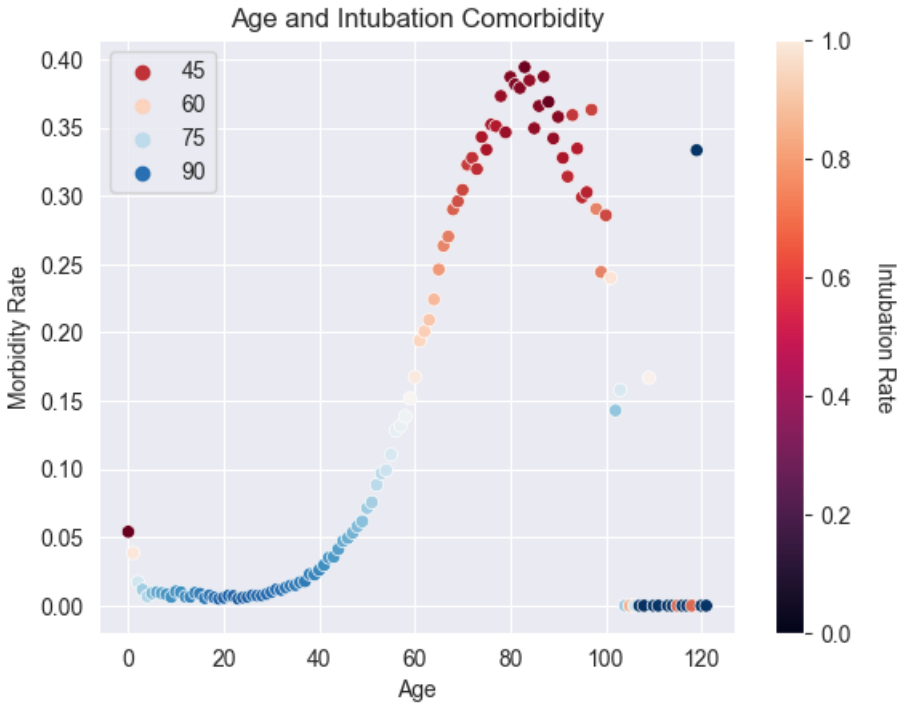


Analysis of Features

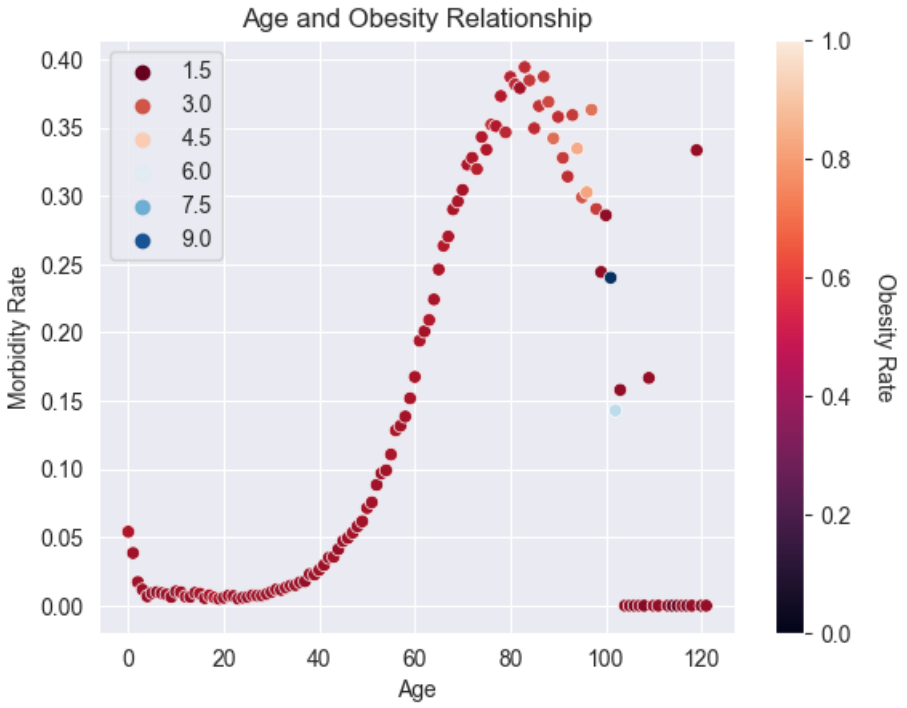
The following cells analyze various features and how these feature interact.

1. Age

```
grouped = df.groupby("AGE")
death_percentages = grouped["DEATH"].mean()
intubation_percentages = grouped["INTUBED"].mean()
ax = sns.scatterplot(x=death_percentages.index, y=death_percentages.values, hue=intubation_p
plt.xlabel("Age")
plt.ylabel("Morbidity Rate")
plt.title("Age and Intubation Comorbidity")
cbar = plt.colorbar(ax.collections[0])
cbar.set_label("Intubation Rate", rotation=270, labelpad=20)
plt.savefig("age_intubation_comorbidity.png")
plt.show()
```




```
grouped = df.groupby("AGE")
death_percentages = grouped["DEATH"].mean()
obesity_percentages = grouped["OBESITY"].mean()
ax = sns.scatterplot(x=death_percentages.index, y=death_percentages.values, hue=obesity_perc
plt.xlabel("Age")
plt.ylabel("Morbidity Rate")
plt.title("Age and Obesity Relationship")
cbar = plt.colorbar(ax.collections[0])
cbar.set_label("Obesity Rate", rotation=270, labelpad=20)
plt.savefig("age_obesity_relationship.png")
plt.show()
```



1. Pregnancy

The following cells suggest that pregnant women, likely due to a variety of confounding factors, are not represented in this dataset for any severe effects of COVID-19.

```
df_pregnant = df[(df["SEX"] == 2) & (df["PREGNANT"] == 1)]
died_pregnant = df_pregnant[df_pregnant["DEATH"] == 1]
if died_pregnant.shape[0] > 0:
    print("There are pregnant women who died.")
else:
    print("There are no pregnant women who died.")
```

```
There are no pregnant women who died.
```

```
if any(df_pregnant["INTUBED"] == 1):
  print("There are pregnant women who were intubated.")
else:
  print("There are no pregnant women who were intubated.")
```

```
There are no pregnant women who were intubated.
```

```
if any(df_pregnant["ICU"] == 1):
  print("There were pregnant women who went to the ICU.")
else:
```

```
  print("There were no pregnant women who went to the ICU.")
```

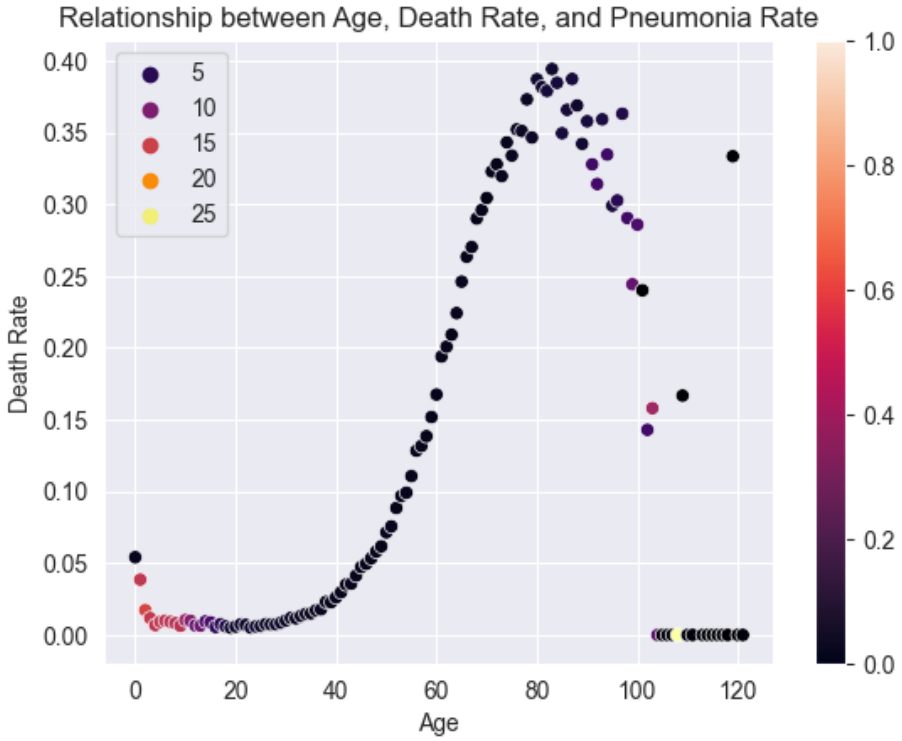
```
There were no pregnant women who went to the ICU.
```

1. Pneumonia

```
# pneumonia, age, and death
grouped = df.groupby("AGE")
pneumonia_rates = grouped["PNEUMONIA"].mean()
death_rates = grouped["DEATH"].mean()
fig, ax = plt.subplots()
scatter = ax.scatter(x=pneumonia_rates.index, y=pneumonia_rates.values, c=death_rates.values
cbar = plt.colorbar(scatter)
cbar.set_label("Death Rate")
plt.xlabel("Age")
plt.ylabel("Pneumonia Rate")
plt.title("Pneumonia and Mortality Rates by Age")
plt.savefig("pneumonia_age_and_fatality.png")
plt.show()
```


```
grouped = df.groupby("AGE")
death_rates = grouped["DEATH"].mean()
pneumonia_rates = grouped["PNEUMONIA"].mean()
fig, ax = plt.subplots()
```

```
scatter = sns.scatterplot(x=death_rates.index, y=death_rates.values, hue=pneumonia_rates.val
plt.xlabel("Age")
plt.ylabel("Death Rate")
plt.title("Relationship between Age, Death Rate, and Pneumonia Rate")
cbar = plt.colorbar(scatter.collections[0])
plt.savefig("age_pneumonia_fig.png")
plt.show()
```



The graphs above would indicate that pneumonia was a much more prominent co-morbid factor in very young people who had COVID-19. Thus, we can understand it more as an age-related factor rather than a true indicator.

1. Sex

Here is a visualization of age and sex comorbidity relationship.

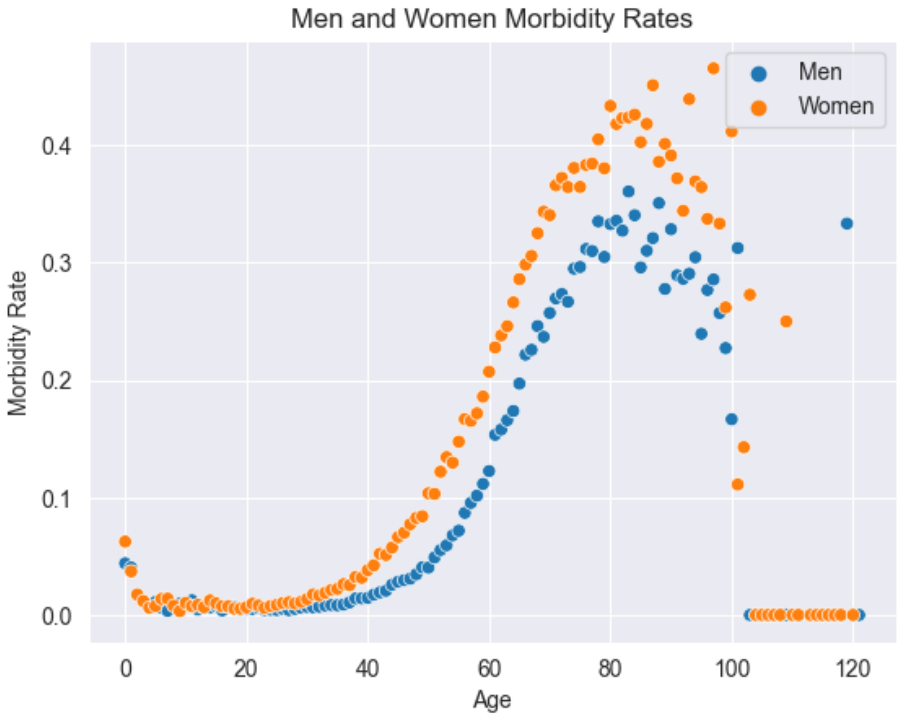
```
# men plot
df_men = df[df["SEX"] == 1]
grouped_men = df_men.groupby("AGE")
death_percentages_men = grouped_men["DEATH"].mean()

# women plot
df_women = df[(df["SEX"] == 2)]
```

```
grouped_women = df_women.groupby("AGE")
death_percentages_women = grouped_women["DEATH"].mean()

# Combine data for men and women
death_percentages = pd.concat([death_percentages_men, death_percentages_women])
sex = ["Men"] * len(death_percentages_men) + ["Women"] * len(death_percentages_women)

# Plot combined data
sns.scatterplot(x=death_percentages.index, y=death_percentages.values, hue=sex)
plt.title("Men and Women Morbidity Rates")
plt.xlabel("Age")
plt.ylabel("Morbidity Rate")
plt.savefig("male_female_morbidity_fig.png")
plt.show()
```



Finally, is interesting to look at differences between the sexes in morbidity rates and their relationship to age and intubation.

```
# men plot
df_men = df[df["SEX"] == 1]
grouped_men = df_men.groupby("AGE")
death_percentages_men = grouped_men["DEATH"].mean()
intubation_percent_men = grouped_men["INTUBED"].mean()
```

```
# women plot
df_women = df[(df["SEX"] == 2)]
grouped_women = df_women.groupby("AGE")
death_percentages_women = grouped_women["DEATH"].mean()
intubation_percent_women = grouped_women["INTUBED"].mean()

# Combine data for men and women
death_percentages = pd.concat([death_percentages_men, death_percentages_women])
intubation_percent = pd.concat([intubation_percent_men, intubation_percent_women])
sex = ["Men"] * len(death_percentages_men) + ["Women"] * len(death_percentages_women)

# Plot combined data
sns.scatterplot(x=death_percentages.index, y=death_percentages.values, hue=intubation_percen
plt.title("Men and Women Intubation and Morbidity Rates")
plt.xlabel("Age")
plt.ylabel("Morbidity Rate")
plt.savefig("male_female_intubation_fig.png")
plt.show()
```

Model Comparison

All the models performed well. The neural network does indeed have impressive performance, but in addition to computationally costly training, it is opaque how to interpret the feature importance. Thus, the random forest, in my opinion, stands out. In return for the training time there it returns a model that performs well and is highly suited to visualization in both prediction and determining feature importance.

Models

1. Random Forest

```
# RF evaluation
y_pred = forest.predict(X_test)
report = classification_report(y_test, y_pred, output_dict=True)
accuracy_list.append(report["accuracy"])
precision_list.append(report["1"]["precision"])
recall_list.append(report["1"]["recall"])
f1_list.append(report["1"]["f1-score"])
```

1. Logistic Regression

```
# logistic regression model
model = LogisticRegression(solver="lbfgs", random_state=42)
model.fit(X_train, y_train)
```

```
C:\Users\edwin\OneDrive - University of Denver\Personal Programming\venv\lib\site-packages\s
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression(random_state=42)
```

```
# LR evaluation
y_pred = model.predict(X_test)
report = classification_report(y_test, y_pred, output_dict=True)
accuracy_list.append(report["accuracy"])
precision_list.append(report["1"]["precision"])
recall_list.append(report["1"]["recall"])
f1_list.append(report["1"]["f1-score"])
```

1. Decision Tree

```
# decision tree
model = DecisionTreeClassifier(max_depth=250)
```

```
model.fit(X_train, y_train)
```

```
DecisionTreeClassifier(max_depth=250)
```

```
# DT evaluation
y_pred = model.predict(X_test)
report = classification_report(y_test, y_pred, output_dict=True)
accuracy_list.append(report["accuracy"])
precision_list.append(report["1"]["precision"])
recall_list.append(report["1"]["recall"])
f1_list.append(report["1"]["f1-score"])
```

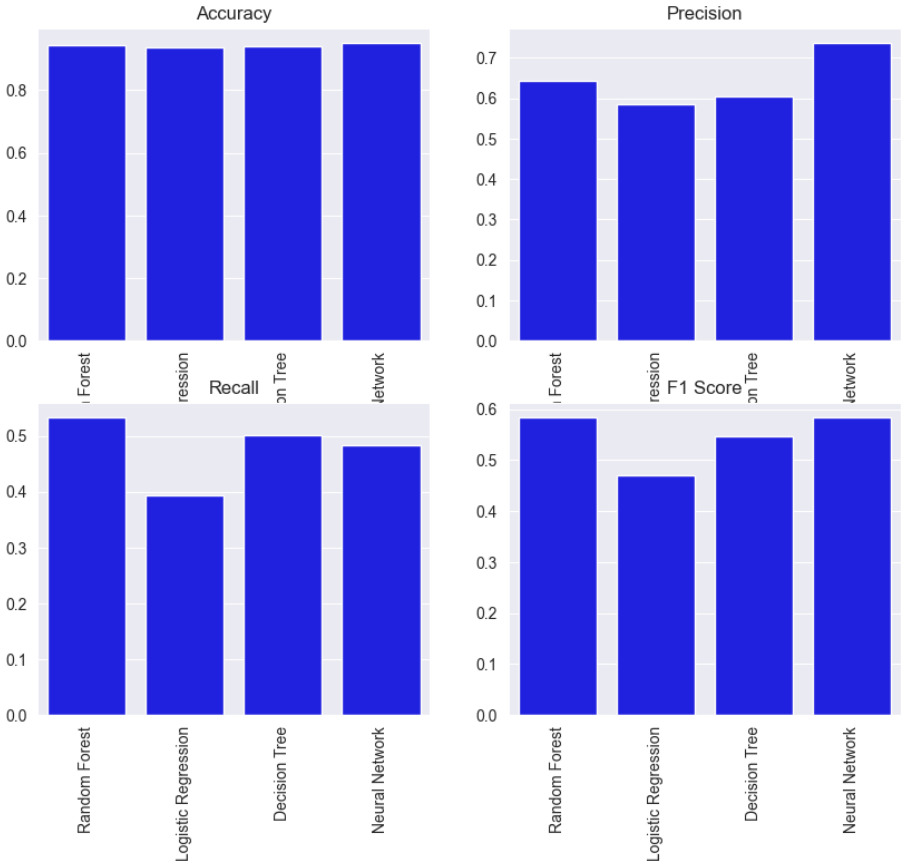
1. Neural Network

```
# NN creation
model = MLPClassifier(hidden_layer_sizes=(10, 10))
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
# NN evaluation
y_pred = model.predict(X_test)
report = classification_report(y_test, y_pred, output_dict=True)
accuracy_list.append(report["accuracy"])
precision_list.append(report["1"]["precision"])
recall_list.append(report["1"]["recall"])
f1_list.append(report["1"]["f1-score"])
```

Comparison

```
# comparison plot
model_names = ["Random Forest", "Logistic Regression", "Decision Tree", "Neural Network"]
num_models = len(model_names)
fig, ax = plt.subplots(2, 2, figsize=(10, 8))
sns.barplot(x=model_names, y=accuracy_list, ax=ax[0, 0], color="b")
ax[0, 0].set_title("Accuracy")
ax[0, 0].set_xticklabels(model_names, rotation=90)
sns.barplot(x=model_names, y=precision_list, ax=ax[0, 1], color="b")
ax[0, 1].set_title("Precision")
ax[0, 1].set_xticklabels(model_names, rotation=90)
sns.barplot(x=model_names, y=recall_list, ax=ax[1, 0], color="b")
ax[1, 0].set_title("Recall")
ax[1, 0].set_xticklabels(model_names, rotation=90)
sns.barplot(x=model_names, y=f1_list, ax=ax[1, 1], color="b")
ax[1, 1].set_title("F1 Score")
ax[1, 1].set_xticklabels(model_names, rotation=90)
plt.savefig("comparison_plot.png")
```


Accuracy
Precision
Recall
F1 Score
Random Forest
Logistic Regression
Decision Tree
Neural Network